PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: **BROOKE, ET AL.**	§	Group Art Unit: **2275**
Serial No.: **09/399,451**	§	Examiner: **QUELER, ADAM M.**
Filed: **09/20/1999**	§	
Title: **"XML SERVER PAGES LANGUAGE"**	§	Atty. Docket No.: **016295.1090**

CERTIFICATE OF MAILING VIA EXPRESS MAIL

PURSUANT TO 37 C.F.R. § 1.10, I HEREBY CERTIFY THAT I HAVE INFORMATION AND A REASONABLE BASIS FOR BELIEF THAT THIS CORRESPONDENCE IS BEING DEPOSITED WITH THE U.S. POSTAL SERVICE AS EXPRESS MAIL POST OFFICE TO ADDRESSEE, ON THE DATE BELOW, AND IS ADDRESSED TO:

HONORABLE COMMISSIONER FOR PATENTS
MAIL STOP NON-FEE AMENDMENT
P.O. BOX 1450
ALEXANDRIA, VA 22313-1450.

Ronald L. Chichester

RONALD L. CHICHESTER REG. NO. 36,765

DATE OF MAILING: 07/17/2003 ~~05/23/2003~~

EXPRESS MAIL LABEL: ~~EV~~ EV337981225US

MAIL STOP NON-FEE AMENDMENT

Honorable Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF PRIOR INVENTION IN THE UNITED STATES TO OVERCOME CITED PUBLICATION UNDER 37 C.F.R. §1.131

Dear Sir:

This declaration is to establish completion of the invention of this application in the United States at a date prior to June 11, 1999, that is the effective date of the prior art publication entitled "EXtensible Server Pages (XSP) Layer 1" that was cited by the examiner.

The person making this declaration is an inventor. The other inventor, David Brooke, is no longer employed by the assignee of the instant application, is outside of the United States, and is thus unavailable to sign the instant declaration.

To establish the data of completion of this application, the following attached documents are submitted as evidence:

- Invention Disclosure Documents

From these documents, it can be seen that the invention in this application were made at least by the date of April 20, 1999 (the day that the invention disclosure was signed and submitted to the legal department of the assignee), which is a date earlier than the effective date of the reference.

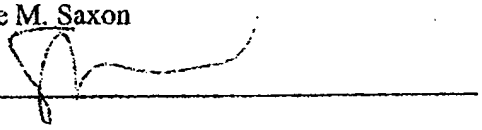
Applicants were diligent in their efforts to file a patent application, and did so within five months from the above-referenced disclosure date.

This declaration is submitted prior to a final rejection.

As a person signing below:

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of the second inventor: Steve M. Saxon

Inventor's signature: 

Date: 7/16/2003

Country of Citizenship: United Kingdom

Post Office Address: ONE DELL WAY, ROUND ROCK
TEXAS 78682, USA

DELL CONFIDENTIAL

INVENTION DISCLOSURE FORM

(Rev. 9/15/98)

INSTRUCTIONS:

- *Make sure all blanks in the form are **completely** filled out. Incomplete forms **will not be processed**.*
- *Have all inventors electronically "sign" the form at the end by simply typing in the name and date in the pertinent blanks at the end of the form (no pen or pencil necessary).*
- *Have two (2) witnesses "sign" in the same fashion.*
- *If submitting drawings with the disclosure, please embed them as a graphic in a Microsoft Word file at the end of the disclosure.*
- *Send completed disclosure **via e-mail** to Benjamin Solomon at Dell Legal.*

***IMPORTANT!** - If you know for a fact that your idea was embodied in a product offered for sale more than a year ago, then please tell us now; otherwise you will have to refund your invention award to the company at a later date.*

INVENTION TITLE:

(Brief and descriptive) XML Server Pages language – a scripting language expressed as an XML vocabulary for the purpose of describing the process of dynamic XML document creation, typically on a web server, but also as a service to other software components.

INVENTORS:

(Must be filled out completely)

1st Inventor Full Legal Name: David Brooke Employee No.: 64329 (EMEA)
Cost Center: 0298-20650 SSN: WL 08 77 96 A Phone Ext.: +44 1344 748752 Bldg.: PK2
Home Address: 1 Westview Rise City: Hemel Hempstead State: Hertfordshire ZIP: HP2 5DQ
Home Phone: +44 1442 266060
Are you a U.S. Citizen? No If no, of which country are you a citizen? United Kingdom
Reporting Director: Gordon Ballantyne
Reporting VP: Maurice Cowey Check here if inventor is non-Dell:

2nd Inventor Full Legal Name: Steve Saxon Employee No.: 103836 (EMEA)
Cost Center: 0298-20650 SSN: NH 77 71 96 B Phone Ext.: +44 1344 748681 Bldg.: PK2
Home Address: 47 Southwold City: Bracknell State: Berkshire ZIP: RG12 8XY
Home Phone: +44 1344 424461
Are you a U.S. Citizen? No If no, of which country are you a citizen? United Kingdom
Reporting Director: Gordon Ballantyne
Reporting VP: Maurice Cowey Check here if inventor is non-Dell:

DEVELOPMENT PARTNER/CONSULTANT:

Was the invention developed in conjunction with a development partner or consultant that contributed to the invention? N If YES, please list here:

DOCUMENTATION

Date of conception: August 1998
Invention first described in: Online language documentation created September 1998 and last updated January 1999 (accessible at http://wotsit/implementation/scripting/scriptml.htm)

Additional/detailed description in: Full language documentation created February 1999 (accessible at http://wotsit/implementation/scripting/xsp.htm)

PRODUCT LINE:

This invention is most closely related to the following product line: *(ONLY ONE WILL BE PROCESSED)*

Dimension_____
Optiplex _____
Inspiron_____
Latitude_____
Servers_____
Workstations_____
Other (e.g. SW, Mfg., BTO) www.dell.com

Code name of Dell Product in which invention is or will be incorporated: No code name – basis for delivery of www.dell.com content in EMEA (to be expanded globally)

FIRST DISCLOSURE, USE OR OFFER OF SALE OF THE INVENTION

PLEASE DO NOT SKIP THIS PART. This information is used to determine Dell's legal rights in the invention.

Has the invention been disclosed outside of Dell? Y___Y_____N
If YES, to whom was this disclosure made? Microsoft Corporation, Object Design Corporation
Was this disclosure made under a non-disclosure agreement (NDA)? Y___Y_____N
IF YES, date of NDA Permanent overall NDA with Microsoft Corp., General NDA with ODI forming part of contract at time of purchase of ObjectStore software in July 1998

Planned date of first offer of sale of product using the invention: Not for sale (if sale has not already occurred)
Actual date of first offer of sale of product using the invention: Not for sale (if sale has already occurred)
Date of first production use of the invention or ship date: February 15th 1999

INDUSTRY STANDARDS / STANDARDS COMMITTEES

Does this invention relate to or incorporate any industry standards? Y_____Y_____N
If YES,

1) Which standard? XML (eXtensible Markup Language) 1.0 published 10th February 1998, XSL (eXtensible Stylesheet Language) current Working Draft (published December 1998)
2) Name of industry standards committee World Wide Web Consortium
3) Is Dell a member of that standards committee?_____Y N___N
4) Name of Dell's representative to the standards committee: N/A

COMPLETE WRITTEN DESCRIPTION OF INVENTION:

Prepare a written description of your invention using the outline below. Just fill in the blank after each topic. Adjust the amount of space for each topic as needed. ***Be sure to include any sketches, diagrams, flow charts, drawings, prints, etc. which will aid in understanding the invention.***

a) THE PROBLEM

Dell's EMEA websites were originally composed as hard-coded HTML and/or ASP pages, containing both data and formatting in a single document.

- The same information was duplicated across many pages to incorporate subtle variations for localization purposes in individual IBUs.
- Adding a translation involves coding a complete new set of marked up pages, rather than just adding data.
- Many pages incorporated content owned by multiple contributors.

- All content was tied to current HTML standards, such that support for new client platforms such as PDAs or WebTV, and more particularly new online markup languages such as WML (Wireless Markup Language for mobile devices such as phones) would require duplication of content.

All of these factors acted to limit Dell Online EMEA's ability to scale the amount of content, the variety of content and the types of content provided to customers, as well as making excessive demands on the organisation.

The problem was characterized as requiring:

- Methods of separating content and behaviour of web content from the data itself
- Ways of storing data in structured, but flexible collections associated with owners and recombining/reusing the data in many different pages, each of which may draw on any number of sources
- Data being the driver, with few scripts – scripts would also not act as rigid templates
- Maximum leverage of XML-based standards
- Techniques for smooth integration of document data and data from dynamic sources (such as Dell IT systems)
- An inheritance mechanism to allow the grouping of 'pages' (no longer hard-coded) into 'classes' and simple derivation of 'sub-classes' of pages from them (e.g. to be able to say this is a www.dell.com page, the Dell home page is a kind of www.dell.com page, and a segment home page is a kind of Dell home page; or this is a product specification page, a notebook specification page is a kind of product specification, and a Latitude specification is a kind of notebook specification).

b) PRIOR METHODS/APPARATUS USED TO SOLVE THE PROBLEM

1. **Commercially available content management systems (Vignette StoryServer, Inso Dynabase, etc.)**:
   These typically use templates or page components that are dynamically populated from SQL databases and recombined into pages using pre-defined templates.
   - These systems generally fit well with highly structured sites having many identically formatted pages, such as a news site.
   - The template structures are generally fixed and not flexible, and there is no by-design inheritance mechanism.
   - The data storage paradigm is based upon filling named slots in the templates, and is thus amenable to tabular storage in a SQL store, but does not fulfil Dell's requirement to use a flexible but structured data format that prioritises the expression of data and its relationships.
   - The template model for such systems is typically based on either Java, or a scripting language such as VBScript or Tcl/Tk.
   - Limited support is typically provided for XML as a data type.

2. **Internet Application Servers (ColdFusion, etc.)**:
   These are primarily targeted at supporting the development of interactive applications, and provide only the necessary basic support for content management.
   - Most of the site template structures are hard-coded as server scripts (see ASP below), often using a mixture of standard HTML tags and proprietary tags that are pre-processed on the server. Once again, each script stands alone, with no inheritance mechanism, although advanced functionality can be placed in separate COM components.
   - Even though tag-based, the scripts for such systems are not well-formed XML but customized HTML, and the separation of form and data is limited.
   - Again, XML can potentially be used in such an environment, but limited to complete source data files.

3. **Web-enabled object/XML databases (ObjectStore/eXcelon, Poet, etc.):**
   These provide a platform for high-performance application development around a flexible repository, but none of the tools.
   - Most application development is left to the purchaser – there are only limited development tools, and no high-level scripting language to provide a content-management framework.
   - The data-modelling capabilities are flexible and well-suited to free-form web content.

4. **Traditional non-Web content management systems (Interleaf, ArborText, TexCel, etc.):**
   These are targeted at generic, media-neutral content management, and are frequently SGML-based, leading to a natural evolution towards XML. They are currently typically deployed for the maintenance of major documentation projects by companies such as Boeing! Each system output will normally be customized (because of special order options), and may be delivered online, on CD, or as print.
   - Separation of data and formatting is thorough, but there is a close mapping between stored data sets and output pages/chapters. Recombination and reuse are supported.
   - The priority is on clearly-controlled document management, rather than on dynamic delivery of flexibly presented and highly-persuasive web content.
   - The working paradigm is based on explicit document assembly, rather than data-driven, script-aided document delivery.

5. **Population of pages using ASP (possibly ISAPI) and SQL, with content selection rules support from personalization/recommendation software components:**
   This is possibly the simplest approach to building an in-house content management system.
   - Again, most of the site template structures are hard-coded, this time natively in HTML, and there is inherently no inheritance mechanism.
   - Most of the data is embedded in the pages, with the 'personalisation' problem being reduced to the population of pre-defined slots with targeted data. Mass customization is possible, but there is little flexibility.
   - The programming paradigm for such a system is locked into Windows Scripting Host-supported scripting languages, typically JavaScript and VBScript. XML can potentially be used in such an environment, but limited to complete source data files.
   - Both the data schema and templates are relatively fixed and slow to adapt.

c) PROPOSED SOLUTION TO THE PROBLEM

Dell's solution rests on several principles:

1. Use XML as the standard underpinning all markup of document components
2. Establish a lightweight server-based scripting framework, expressing all language features in an XML-compliant vocabulary, so that all system documents are well-formed, and scripts are simply another kind of data entity in the system
3. This framework must make maximum use of standardized XML technology, in other words it should not duplicate XSL or XML Schema capabilities, but supplement them and coordinate their use.
4. The system should share common syntax with XSL where possible (e.g. for control structures), leveraging the XML namespace mechanism for clarity
5. Leverage 2, 3 and 4, so that script tags and other XML vocabularies are capable of being freely interleaved.
6. Make the system restructure and represent the site by responding directly to changes in data
7. The system should maximize the reusability of all system components: XML data documents, scripts, etc.

The Dell XSP scripting engine was developed according to these principles, and supports key concepts from object-oriented software development, adapted to an XML grammar.

(i) **XML-based scripts**

Scripts in XSP are used to generate classes of pages, and are initiated by a request to a URL naming the script and supplying parameters, or via a COM interface to the scripting engine.

The XSP script itself is an XML document, the inputs to the script are generally one or more XML documents, and the output of a script is itself an XML object as defined in the W3C DOM. The output may be a well-formed (XHTML) document that is browser-ready, or it may be expressed in another XML vocabulary that is ready for use in a data-interchange process with another system, or for formatting and delivery to an HTML or non-HTML client.

XSP scripts are interpreted by the engine as a set of subroutines that create XML structures in the output tree. These may be created by querying and transforming one or more XML data sources (using XQL/XSL), or by direct inline coding of desired output (template-style – generally deprecated, but possible where required).

Facilities also exist for creating and accessing named and scoped variables, exercising flow control, instantiating and accessing COM objects, and creating server-side XML data islands with script scope. Where possible, control structure syntax is shared with XSL, but placed in a different XML namespace to clarify which tier of the application will handle execution.

(ii) **Script inheritance**

Execution of an XSP script commences with a subroutine named "main", mirroring C++ notation, and proceeds through the steps defined, with a call mechanism providing the ability to transfer control. Each XSP script may name a "base" script through the use of an attribute on the root element, so that a given script can implement alternate subroutines (subclassing), or additional subroutines – thus deriving from, or extending, another class of script. A script may also access routines directly from another script, using it as a subroutine library.

In typical implementations, all core functionality to support the structure of the output tree, whether it is an XML-EDI message, or an XHTML document, will be implemented in low level "base" scripts. "User" scripts will provide simple extensions to these, and the XML data sets referenced will determine the final output tree.

This powerful mechanism results in a very small number of compact scripts being able to support very large websites, with a rich variety of actual document formats.

d) DRAWING, SKETCH

An online demonstration of XSP scripting has been provided at:

http://doodah/xsp/inventors.xsp

This fully-commented script derives from the **inventor-base.xsp** script in the same directory, and uses a single xml file as a data source, creating a server-side data island to support repeated querying of the single source without re-parsing the file.

Browsing the same URLs with an additional parameter, like this

http://doodah/xsp/inventors.xsp?target=raw

will remove formatting and deliver the script output as pure XML (easiest to view with IE5).

The xml source file can also be browsed directly at:

http://doodah/xsp/data/inventors/xsp.xml

All major language features described in this document are demonstrated in this very brief example, with the result that it is not necessarily a good example of best practice!! The source for the two scripts is also attached with this document.

File contents

inventors.xsp

```
<xsp:script base="inventor-base.xsp" xmlns:xsp="uri:xsp" xmlns:xsl="uri:xsl" xmlns:doc="uri:doc">
<!-- This script extends inventor-base.xsp -->

<!-- Over-ride contents implementation in the base script -->
	<xsp:sub name="contents">
		<!-- Ask XSP to get the list of inventors as a server-side data island and hold onto it for script lifetime -->
		<xsp:xml name="listofinventors" src="/xsp/data/inventors/xsp.xml" />

		<!-- Use XSP to query the data island, with a query attribute to retrieve only the inventors (not managers) -->
		<xsp:query src="#listofinventors" query="//inventors">
			<!-- standard XSL root rule to start stylesheet processing of this document fragment -->
			<xsl:template match="/">
				<!-- wrap in doc:root tags with doc namespace to ensure well-formed-ness in the output document-->
				<doc:root xmlns:doc="uri:doc">
					<doc:heading level="3">Inventors</doc:heading>
					<!-- create a pseudo-table in our selected non-HTML tag vocabulary -->
					<doc:table width-rel="100" spacing="2" padding="3">
						<!-- write out the headers -->
						<doc:tablerow>
							<doc:tablecell align="left" colour="silver">Name</doc:tablecell>
							<doc:tablecell align="left" colour="silver">Title</doc:tablecell>
							<doc:tablecell align="left" colour="silver">Badge Number</doc:tablecell>
							<doc:tablecell align="left" colour="silver">Social Security Number</doc:tablecell>
						</doc:tablerow>
						<!-- now ask XSL to process the inventors, ordered by badge number -->
						<xsl:apply-templates select="//inventor" order-by="inventor/badgeno" />
					</doc:table>
				</doc:root>
			</xsl:template>
			<!-- back to XSP to call base subroutine to import the default rules -->
			<xsp:call href="#core-templates" />
			<!-- XSL again to provide a special template for an inventor, outputting selected fields in the desired order -->

			<xsl:template match="inventor">
				<doc:tablerow>
					<doc:tablecell><xsl:value-of select="name" /></doc:tablecell>
					<doc:tablecell><xsl:value-of select="title" /></doc:tablecell>
					<doc:tablecell><xsl:value-of select="badgeno" /> 
						(<xsl:value-of select="region" />)</doc:tablecell>
					<doc:tablecell><xsl:value-of select="SSN" /></doc:tablecell>
				</doc:tablerow>
			</xsl:template>
		</xsp:query>

		<!-- Just for fun, now use XSP to query the same data island a second time for managers and not inventors -->
		<xsp:query src="#listofinventors" query="//managers">
```

inventor-base.xsp

```
<xsp:script xmlns:xsp="uri:xsp" xmlns:xsl="uri:xsl" xmlns:doc="uri:doc">
<!-- Base script -->
<!-- This script will not normally be accessed directly by users -->
<!-- If you do run this script directly it will just output the contents of the XML source as a single string! -->

<!-- Default XSP entry point -->
	<xsp:sub name="main">
		<!-- Insert any standardised output here -->
		<!-- Then call the contents routine - probably implemented elsewhere -->
		<xsp:call href="#contents" />
	</xsp:sub>

<!-- Provide a default XSP contents routine - normally implemented in derived scripts -->
	<xsp:sub name="contents">
		<!-- Ask XSP to query the document -->
		<xsp:query src="/xsp/data/inventors/xsp.xml">
			<!-- Now XSL kicks in - just provide a default rule -->
			<xsl:template match="/">
				<!-- wrap in doc:root tags with doc namespace to ensure well-formed-ness -->
				<doc:root xmlns:doc="uri:doc">
					<!-- ask XSL to process all nodes -->
					<xsl:apply-templates select="*"/>
				</doc:root>
			</xsl:template>
			<!-- Now call an XSP routine to import the default rules before invoking XSL -->
			<xsp:call href="#core-templates" />
		</xsp:query>
	</xsp:sub>

<!-- This XSP routine provides the two XSL default rules whenever needed -->
	<xsp:sub name="core-templates">
		<!-- output all text nodes as is -->
		<xsl:template match="textNode()">
			<xsl:value-of />
		</xsl:template>
		<!-- ensure XSL recursion continues in the absence of better matching rules -->
		<xsl:template match="*">
			<xsl:apply-templates />
		</xsl:template>
	</xsp:sub>

</xsp:script>
```

xsp.xml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<inventorlist>
        <managers>
                <manager role="dir">
                        <name>Gordon Ballantyne</name>
                        <title>Director of EMEA Online</title>
                </manager>
                <manager role="vp">
                        <name>Maurice Cowey</name>
                        <title>Vice President of HSB for EMEA</title>
                </manager>
        </managers>
        <inventors>
                <inventor ID="I001">
                        <name>David Brooke</name>
                        <title>Internet Development Manager</title>
                        <SSN>WL 08 77 96 A</SSN>
                        <region>EMEA</region>
                        <badgeno>64329</badgeno>
                </inventor>
                <inventor ID="I002">
                        <name>Steve Saxon</name>
                        <title>Technical Architect</title>
                        <SSN>NH 77 71 96 B</SSN>
                        <region>EMEA</region>
                        <badgeno>103836</badgeno>
                </inventor>
        </inventors>
</inventorlist>
```

DECLARATION:
The invention described in this invention disclosure is submitted pursuant to my Employment Agreement with Dell Computer Corporation.

SIGNATURES OF INVENTORS:

Inventor(s), please sign your full name(s) and enter the date below:

(1) David Brooke Date: 4/20/1999

(2) Steve Saxon Date: 4/20/1999

(If there are more than 2 inventors, please add more signature lines as appropriate.)

DECLARATIONS BY AND SIGNATURES OF TWO WITNESSES:

Witnesses, please sign and date below:

WITNESS 1
This invention was first explained to the undersigned by the inventor(s) on the 16 day of April, 1999 I understood the explanation given by the inventor(s).

John Wheeler Date: April 20, 1999
Signature of Witness 1

WITNESS 2
This invention was first explained to the undersigned by the inventor(s) on the 16 day of April, 1999 I understood the explanation given by the inventor(s).

John Brownlee Date: April 20, 1999
Signature of Witness 2